Using Implicit Calls to Improve Malware Dynamic Execution

Mourad Leslous, Jean-François Lalande, Valérie Viet Triem Tong

Using Implicit Calls to Improve Malware Dynamic Execution

M. Leslous J.-F. Lalande V. Viet Triem Tong

INRIA/CENTRALESUPELEC research project Cidre – INSA CVL

France

Dynamic Analysis: The Need for Malware Triggering

- Malware evade dynamic analysis
- Wait for: a specific time, an event, C&C command
- If suspicious code is triggered
  $\longrightarrow$ Capture dynamic behavior

GroddDroid: a Gorilla for Triggering Malicious Behaviors

- Locates the suspicious code
- Builds an interprocedural control flow graph
- Finds a path that reaches the suspicious code
- Instruments & Executes APK

Problem: Execution paths May Use Implicit Calls

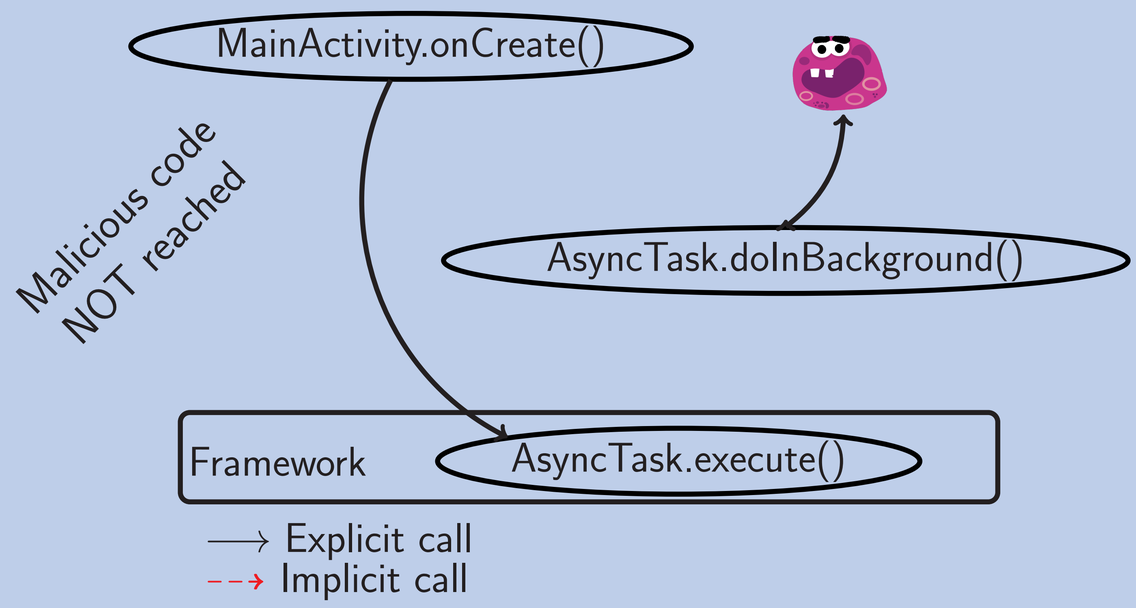
- Alarm Controller (com.al.alarm.controller)
- SHA256: 03c32aca9f894b8a7263e65fd8845a53618f1131877ccb818d018e2d07e2f596
  - Decrypts and loads code dynamically
  - Installs malicious apps
  - Displays unwanted ads
- Malicious code execution path contains an implicit call:
  AsyncTask.execute() -·-> AsyncTask.doInBackground()
- GroddDroid cannot find such a path because of implicit calls

Challenge

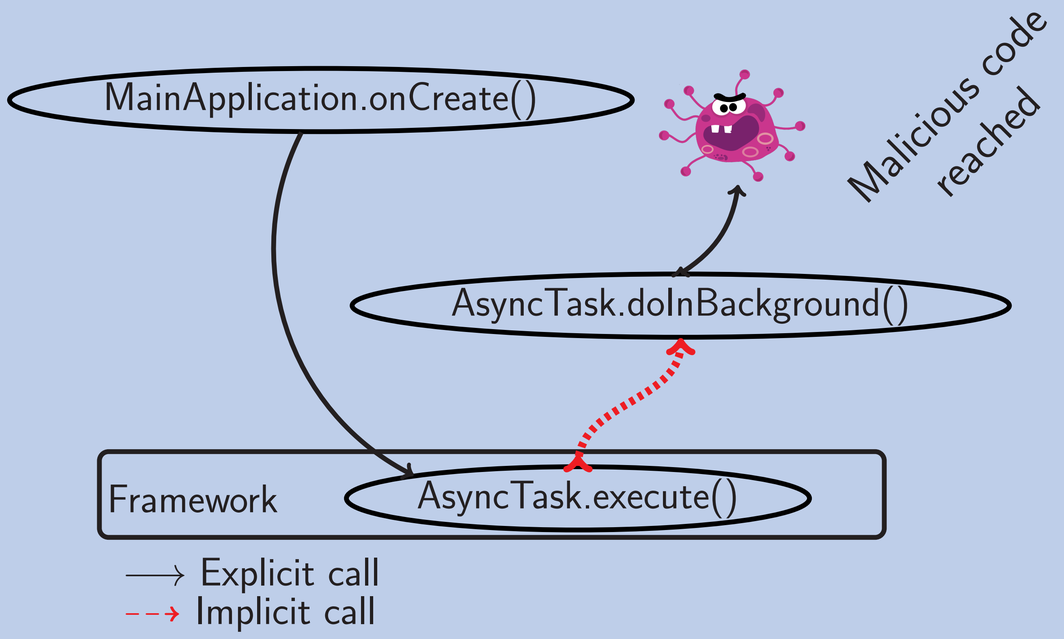
Discover execution paths with implicit calls to reach suspicious code

Including Implicit Calls into CFGs

- Without implicit calls
- Suspicious code execution rate for this sample: 37.5%



- Use pairs of *registration-callback*
- Suspicious code execution rate for this sample: 87.5%



Executing Malware

- Program designed and implemented
- Experiments on large sets of malware
- Measure malicious code coverage with and without implicit calls
- Online malware analysis platform

Contact: mourad.leslous@inria.fr